Overall Introduction of the System

As showing in the following flow chart, the system including four main sections, voting stage one, segmenting, voting stage two, and merging. The video is input as a string of relative address of the frames, and the output including two parts, a summary of the video showing in jpg formatted collage and gif formatted file. And for saving of the computational cost, fifteen 480*640 frames were extracted with a fixed distance from the input video.

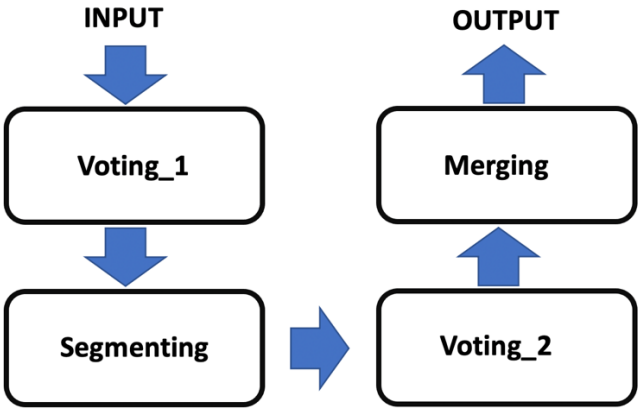

Figure_1 system flow chart

For the first stage Voting_1, we assumed that if a frame that is similar to the previous frame, it is having redundant information; in other words, frames having a large difference from the previous frame indicating they are frames with important information. In this stage, the system tried to calculate the feature difference between the adjacent frames. The feature we were considering including color, edge, key points, CNN features, for generalizing proposes, the system is taking all the features into account.

On the code aspect, four voting functions vote for each frame and one merging function that sums all the voting together to get the top three frames (Figure_2.1).

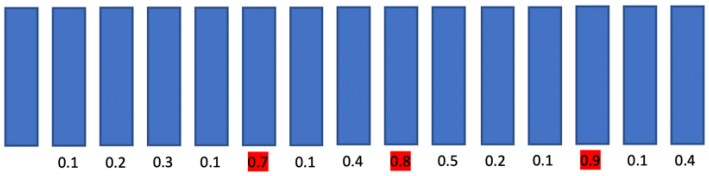

Figure_2.1 Voting_1

And then for the second stage Segmenting, the system basically segments the frames list into four sections based on the three top frames from the last stage.

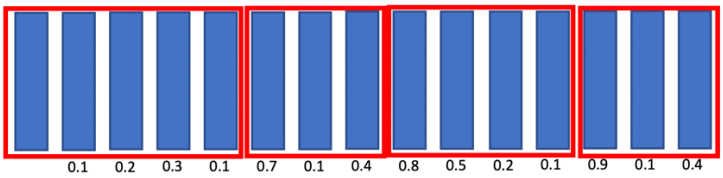

Figure_2.2 Segmenting

And for the next stage, each section will be sent to the function that tries to extract a frame that represents the section best. Similarly, by calling the voting function mentioned in stage one, the sum of the difference of a certain frame was calculated by comparing with the rest frames in the section. The frame with the least total difference is chosen as the representative frame of the section.

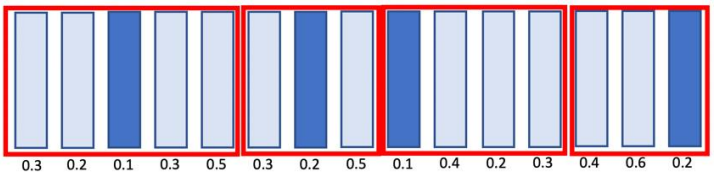

Figure_2.3 Voting_2

Lastly, with the four frames from the previous steps, a collage and a GIF were made in the merge stage.

Method in Details

1&2. Voting based system to divide the video frame sequence into four parts.

The task is performed by combining scores from four different methods that provide us scores on how different two adjacent frames are in the video. The scores from these methods are added together to get the most dis-similar pair of frames which helps in dividing the video frame sequence in to four unique sections. The methods used to obtain these scores are:

Keypoint_based_vote(frames)

Edge_based_vote(frames)

Grayscale_based_vote(frames)

CNN_based_vote(frames)

Keypoint_based_vote(frames)

This function takes the sequence of frames as an input and provides a list of scores on how different adjacent frames are. The method uses local binary pattern (LBP) to find regions of common textures and next computes the histogram for them. These two histograms are compared with each other using Kullback Leibler Divergence, that gives a score between 0 – 1, on how different the scores are (where a score of 0 represents same images). The process of computing the KL divergence score is repeated for every adjacent frame and these scores will be used to get which two frames are least similar. The scores are normalised using min-max normalization. This would in turn help in dividing the sequence frames into four parts and getting the most important frames.

Edge_based_vote(frames)

The method performs edge-based similarity between adjacent frames to find the least similar frames. This approach uses canny edge detector to find edges in the frames and compares them with the help of structural similarity method from skimage library which gives a score between 0 and 1. The score obtained shows how similar two frames are, however to find how different the frames are, the similarity score is subtracted from 1. The structural difference score is normalised between 0 and 1 using min-max normalization. The normalised structural difference score for each pair of frames is returned as an output from the function.

Grayscale_based_vote(frames)

This technique is the simplest of all methods used. It first converts the frames into their grayscale image and uses the structural similarity function from skimage library to find the similarity score between adjacent frames. The similarity score is then converted to dissimilarity score by subtracting similarity score from 1 and normalised using the min-max normalization.

Object_based_vote(frames)

This technique was applying the SSD pre-trained CNN model from TensorFlow, the model was used for object detection. And the output was used in two aspects:

- general object comparison

- zoom motion detection

For the general object comparison, after achieving the result form model, the top 10 objects that having the highest score was listed for each frame. By calculating the similarity between two lists of the objects using difflib.SequenceMatcher(), a score will get based on the similarity.

To make the system even more generalized on video segmentation, it also detects camera zoom-in and zoom-out motion based on the object size. By considering this, the system is more sensitive to Close-Up and long-shot frames.

3.Voting based system to get the representative frames.

After splitting the whole video into multiple parts, the next thing we are going to do is to find the most important frames in each part. To do that we will continue using the 4-voting method to compare the differences between each frame. And since in this case, the frames should represent that part of the video the most, we are going to select the frame that has the least vote among them all. While processing the vote to gather the data it is also different from what we do when splitting the video. Instead of comparing the frame next to the selected frame, we are going to compare the current frame to all the other frames in this part and add all the votes up together and assign a value to the current to represent the differences. In the end, we will select the frame that has the leaset differences among them all so that this frame can represent this part the most.

4.Merging the selected frames into collage and GIF.

GIF

As we have got the important frames from the video, now we can use them to prepare the transition frames for the GIF. The motivation for the transition used came from the video provided in the Ed discussions, where transition from one frame to the other was done by fading the first the image and making the second image appear. The function used to create the transition Gif_frames(frames, fps) takes the important frames and fps – frames per second as the input and displays the gif frames and saves them in the current directory.

The function uses the equation to produce the transition frames:

new_frame = (1 - alpha) * first_frame + alpha * second_frame

where alpha is a balancing coefficient whose value changes from 0 to 0.9.

In the beginning, the first frame is prominent meaning first frame is better visible than the second image and as the value of alpha increases above 0.5, second image starts becoming more visible. According to

the code, the transition take effect for one second with a default of 10 frames per second, meaning if there are four important frames, then the gif will last for 4 seconds with 40 frames.

Collage

The mechanism used was similar to the GIF, the equation for the transition areas is showing as following:

overlap_area_pixel = (1 - alpha) * first_frame_ pixel + alpha * second_frame_ pixel

The alpha is a distance related parameter, it evenly increased from 0 to 1 as the calculating pixel move close to the second frame. The function inputs four frames at a time, it first merges two sets of frames horizontally, and lastly merge them vertically into a 2 by 2 collage.

Sample Outputs

Figure_3.1 collage_1

Figure_3.2 collage_2

Figure_3.3 collage_3

Figure_3.4 collage_4

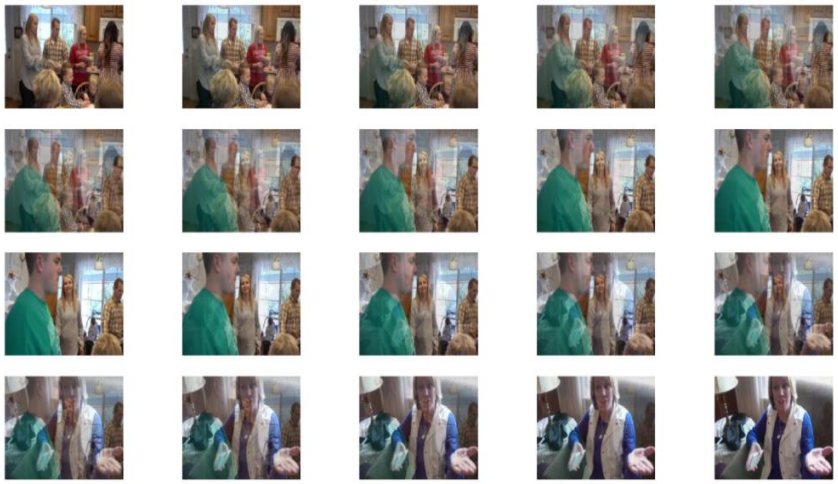

Figure_3.5 transition frames for gif